ESP8266

I2C

User Guide

Version 0.2

Espressif Systems IOT Team

http://bbs.espressif.com

Table of Contents

1. Overview

As an interface of I2C master, ESP8266 can control and read the I2C slave (e.g., most of the digital sensors).

Each GPIO pin could be configured as open-drain . In this way, GPIO interface could function as I2C data or clock.

Meanwhile, there are pull-up resistors within the chip.

The wave forms of SDA and SCL lines used in I2C are created by GPIO simulation. SDA accesses data with SCL high. Both the high and low level of SCL are kept at 5us and the clock rate of I2C is about 100KHz.

2. I2C master interface

2.1. Initialisation

`i2c_master_gpio_init`: initialise the GPIO hardware.

Details are as follows:

(1) Choose the function of pin and configure it to GPIO.

(2) Configure GPIO to the open-drain mode.

(3) Initialise the SDA and SCL to high level.

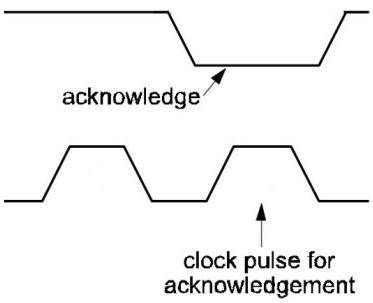
(4) Enable the GPIO interrupt and reset the slave device.

`i2c_master_init(void)`: reset the slave status

2.2. Start the I2C

`i2c_master_start(void)`: start condition for the master to start I2C.

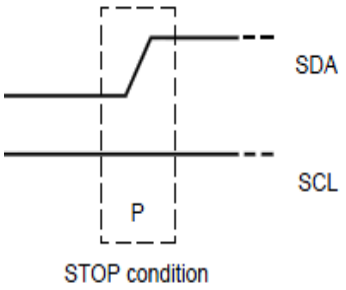
The start condition is shown below:



2.3. Stop the I2C

`i2c_master_stop(void)`: stop condition for the master to stop I2C.

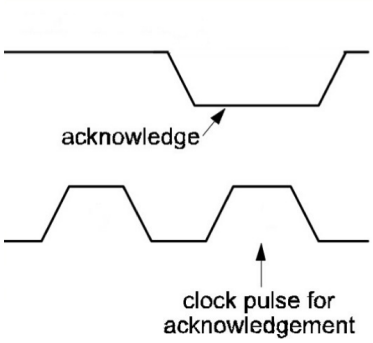
The stop condition is shown below:



2.4. I2C master's response to ACK

`i2c_master_send_ack(void)`: set the I2C master's response to ACK.

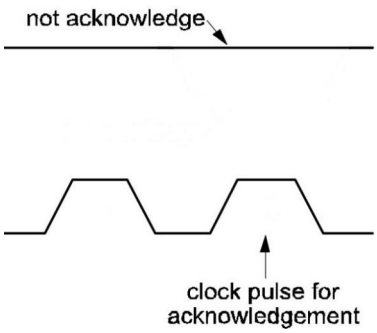
The response is shown below:



2.5. I2C master's response to NACK

`i2c_master_send_nack(void)`: set the I2C master's response to NACK.

The response is shown below:



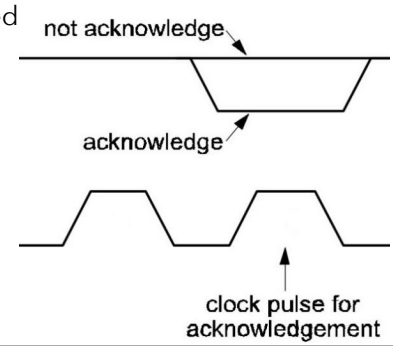
2.6. Check the I2C slave response

`bool i2c_master_checkAck(void)`: check the slave response status.

Return value:

TRUE: slave acknowledge

FALSE: slave not acknowled



2.7. Write data into the I2C bus line

`i2c_master_writeByte(unit8 wrdata)`: write data into the I2C bus line.

Parameter: 1 byte of data

Notices:

1. Data will be sent in descending order of the bit order.
2. The slave address and data can be sent.

2.8. Read data from the I2C bus line

i2c_master_readByte (void): read a byte from the SPI slave.

Return value:

the one-byte data read

3. Example

Refer to *IOT_Demo* from *esp_iot_sdk*. The example is shown below:

```
LOCAL bool ICACHE_FLASH_ATTR
user_mvh3004_burst_read(uint8 addr, uint8 *pData, uint16 len)
{
    uint8 ack;
    uint16 i;

    i2c_master_start();
    i2c_master_writeByte(addr);
    ack = i2c_master_checkAck();

    if (!ack) {
        os_printf("addr not ack when tx write cmd \n");
        i2c_master_stop();
        return false;
    }

    i2c_master_stop();
    i2c_master_wait(40000);

    i2c_master_start();
    i2c_master_writeByte(addr + 1);
    ack = i2c_master_checkAck();

    if (!ack) {
        os_printf("addr not ack when tx write cmd \n");
        i2c_master_stop();
        return false;
    }

    for (i = 0; i < len; i++) {
        pData[i] = i2c_master_readByte();

        if (i == (len - 1))
            i2c_master_send_nack();
        else
            i2c_master_send_ack();


    }

    i2c_master_stop();

    return true;
} ? end user_mvh3004_burst_read ?
```